摘 要

新一代视频编码标准HEVC于2013年正式发布。与前代标准H.264/AVC标准相比，HEVC标准在编码压缩率上取得了大幅提升，但同时也增加了约70%的编码时间。这一耗时的增加主要源自于HEVC标准中更加复杂的图像划分方式与计算过程。正是由于其低下的运行速度，HEVC目前仍难以在市场中被大规模使用。以HEVC标准测试模型为参照，基于CPU-GPU的多核异构平台，利用CUDA编程平台对HEVC中的复杂运算进行了并行优化，充分利用了目前家用电脑中日渐常见的GPU计算资源。

对于HEVC中的两大主要耗时模块：帧内预测模块化和变换量化模块，分别选取其中核心功能函数进行并行算法的设计、实现与测试。在帧内预测模块中，对哈达玛失真函数实现并比较了2种优化方法，加速比达到919%；对预测值计算函数实现了并行优化，加速比达到170%。在变换量化模块中，对离散余弦变换函数设计并比较了2种优化方法，加速比达到811%；对离散正弦变换函数实现并行优化，加速比达到136%。最终试验结果表明，在模块和编码器层面，对帧内预测模块整体提高运行速度22%，对变换量化模块整体提高运行速度25.8%，对编码器整体整体提高运行速度19.7%。

## 课题背景

作为信息时代最重要的信息载体之一，视频具有直观感受性、高效传播性和广泛应用性等优点。近年来，随着视频技术以及Internet技术的迅猛发展，视频应用已经从传统的电视技术转向了数字视频广播、便携摄像以及移动无线视频等领域，更加深刻地改变和影响着日常生活。据统计，视频流数据再2015年占到了整个互联网流量地90%左右，互联网几乎已经成为视频网了[1]。

另一方面，虽然网络网络带宽和传输能力增加迅速，但仍远不能满足海量视频数据的传输和存储要求。以常见的1280×720视频序列为例，如果三个色度分量都用8bit表示，则一帧图像的原始数据量约为22Mbits。假如传输帧率为25fps，视频时长为一个小时。则整个视频的大小将达到约2Tbits，这是当前网络带宽和存储设备都难以承受的。因此如何高效地压缩视频，一直以来都是视频处理相关技术研究的核心课题。从20实际80年代开始，国际电信联盟电信标准化组织（ITU-T）和国际标准化组织（ISO）/国际电工委员会（IEC）就一直致力于视频编码技术的标准化工作。由视频编码专家组（VCEG）和动态图像专家组（MPEG）先后推出了的H.26x系列和MPEG系列等视频编码标准。其中，在2003年发布的H.264/AVC标准由于具有良好的视频压缩率、网络适配性和防出错能力，因此被广泛应用于视频存储、流媒体等领域，成为目前市场占用率最高的视频编码标准之一[2]。

然而，随着目前高清、超高清视频（4K，8K）的逐渐普及，以往以H.264为代表的视频编码标准收到了越来越大的挑战，主要表现为视频编码效率的提高跟不上视频数据量的增加。为了进一步应对这一趋势，2010年，VCEG与MPEG合作组件了视频编码联合小组（JCT-VC），致力于研发新一代“高效视频编码”（High Efficiency Video Coding，HEVC），并在2013年1约正式发布了该标准。与前一代主流编码标准H.264/AVC相比，在保证相同图像质量下，HEVC可以减少约50%的码率[3]。但是，HEVC为了获得优异的压缩而造成的超高的计算复杂度对其推广产生了不小的影响。2018年，HEVC的市场占有率从3%提高为9%，仍远低于H.264的81%。因此，如何提高编码效率成为HEVC相关研究的热门方向。

近年来，随着GPU（Graphics Processing Unit）技术的高速发展，GPU已经不仅仅被用于图像渲染，而被更广泛地用于多种不同地计算任务，包括机器学习、计算机仿真和通信技术等等。GPU在并行计算方面相比于CPU具有明显优势，这主要源于GPU与CPU在处理任务方式上地区别。GPU专为复杂的数学和几何运算而设计，其运算核心由大量较小而高效的核心组成，而CPU一般由一个或几个专门顺序执行的串行处理而优化的核心组成。2006年，NVDIA公司推出了针对其旗下GPU产品的编程平台CUDA（Computer Unified Device Architecture，统一计算框架）。随着这一编程环境的出现和发展，GPU的通用计算编程实用性大大提高，复杂度不断降低，使得GPU真正成为了一种实用的通用并行计算资源。这也使得利用CUDA平台来对HEVC中的相关并行计算进行优化加速成为可能。

## 国内外研究现状

目前关于并行视频编码主要的两种大的研究方向，分别是是帧内并行和帧间并行。帧内并行主要是指使用多线程来处理同一帧之中的不同区域。主要应用于实时编解码领域，使用的线程数较少。这种并行方式的规模是不随着视频长度而变化的，在实际使用中，如果不大量牺牲压缩率，线程数难以超过几十个。这一缺陷限制了其加速的上限。

在帧内并行研究中，运动估计是计算复杂度最高的函数之一。而其中又因为全搜索算法的可并行度较高，所以尝试将其通过GPU加速的相关研究也很多。参考文献[1]提出了一种高效的4x4块级并行算法，这一算法将ME算法分解为5个步骤，最终获得了较好的并行度以及较低的CPU-GPU通信量。参考文献[2]实现了与[1]类似的全搜索算法，达到了99%的加速比，但是同时牺牲了率失真和比特率。参考文献[3]在三种平台上实现了全搜索算法，并且显示出GPU具有最佳的性能提升。参考文献[4]在GPU上实现了异步通信机制，当GPU在运行ME操作时，CPU可以执行其他操作，比如文件读写等等。文献[5]是基于宏块的并行计算，并提出了一种基于CUDA的H.264运动估计算法。 采用的策略分为三步实现，第一步计算4×4块的SAD(SumofAbsolute Difference) 值，第二步计算 4 ×8，8×4 等其余划分块的SAD值，而这些分块的值都是由之前4×4块的SAD值合得到的。最后一步计算比较每个分块SAD的最小值。其策略的每一步计算都是在CUDA上由kernel函数实现。这种做法平均减少 93.6% 的运动估计时间。文献[6]用CUDA实现了smpUMHexagonS快速运动估计算法，提出了把当前帧划分为多个tiles后，每个tile由单一GPU线程处理，不同的tiles在分别独立的线程上并行处理。结果显示码率最大增加12%和质量下降0.4dB，图像质量的下降取决于测试序列的内容和tile的划分长度。

文献[7]在CUDA平台上提出了一个高效的并行执行的运动估计算法，使用一个或两个GPU执行运动估计，而其他部分的软件保持不变。结果显示在使用NVIDIA Geforce GTX680和NVIDIA Geforce GTX690的显卡条件下，该混合编码器处理高清视频序列时，比正常未改变的编码器提高高达42%的速度，而且对视频比特率和质量的影响很小甚至可以忽略不计。文献[8]提出了一种编码树单元CTU级的并行方法。首先提出一个高度可扩展的CTU级的并行运动搜索算法，其中并行的CTU块可以设置为任何尺寸来适应不同分辨率的视频序列和可变的硬件配置。然后运动搜索范围可以在运动强度的基础上自适应地调整。另外，从GPU返回的运动估计的信息可以在快速模式选择中被CPU所用。结果表明，该方法比仅使用CPU实现 HM-10.0 降低了高达73%的复杂度，并且编码性能的损失可以接受。文献[9]提出了一种基于GPU的多层次的并行运动估计方案，包括四层：编码树单元(CTU)，预测单元(PU)，运动矢量(MV)选择和指令优化。在PU层，各种尺寸的PU的代价不是通过类似文献[5]的渐进的代价合并，而是是通过一个SAD查找表得到。而且在运动矢量选择时，通过GPU的选择指令避免了条件分支。结果表明，该方案可以充分利用GPU实现比使用快速运动估计的 HM-10.0超过90倍的加速。

帧间并行主要是指将一个视频切割成多个小段来分别进行处理。主要的好处是可以随着视频长度的变化来延拓，但是同时也会在高并行度下牺牲编码效率。这主要是因为切割视频进行编码会造成每一段都需要插入关键帧，从而造成了整个视频的压缩率变低。参考文献[12]尝试在natural locations来方式关键帧.也就是在本来就需要设置关键帧的地方进行切割，这样就变相减少了额外的切割视频带来的损失。参考文献[13]指出前人工作的主要缺陷是并行的粒度较粗，也就是每一个worker上放几千帧去处理。而其中提出的ExCamera每一个worker上只有很少的几帧。文献[13]中尝试使用数千个AWS Lambda微服务器来进行并行运算，每一个服务器处理4帧图片，获得了较好的实验结果。

## 本文研究内容

本文基于HEVC测试模型（HEVC Test Model，HM）进行研究，主要针对其中的帧内预测模块进行并行优化。对于帧内编码模块中最为耗时的角度预测、率失真函数、变换函数分别提出了并行算法，在CUDA上加以实现，最后编译为静态链接库整合进HM整体代码中。并且针对不同的视频测试序列，对优化后的帧内编码模块进行了测试和分析。最终能较为显著地提高HEVC编码器的bianma效率，并符合并行算法的理论设计预期。

## 论文结构

本文主要内容如下：

第一章首先介绍了视频编码技术的发展历程的目前面临的挑战，然后介绍了新一代高效视频编码标准HEVC的特点，以及其在广泛使用中面临的问题。之后对GPU硬件和编程平台CUDA的发展做了简单介绍，并指出了利用CUDA平台对HEVC进行并行优化的可行性。

第二章中，结合HEVC编码标准和相关理论，详细介绍HM的整体架构和设计细节，对其中核心的编码树结构、帧间预测、帧内预测模块进行分析。然后对CUDA编程平台的软硬件架构和使用方法做进一步说明。

第三章主要针对帧内编码部分进行研究，对角度预测、失真函数、变换函数的流程进行分析，并提出针对每一个算法相对应的并行优化方法。并简单介绍将CUDA代码编译为静态链接库，并整合进HM代码的方法。

第四章主要对优化后的HM代码在不同的视频序列下进行测试，并结合算法的理论并行度对实验结果进行分析。

第五章主要对全文工作进行总结，指出其中不足指出，对进一步可能的改进方向进行展望。